N89-15557

Issues in Management of Artificial Intelligence Based Projects

P. A. Kiss
The BDM Corporation
950 Explorer Boulevard
Huntsville, AL 35806

Dr. Michael S. Freeman
Systems Engineering Division
Systems Analysis and
Integration Laboratory
Marshall Space Flight Center AL

1.0 Abstract

Now that AI is gaining acceptance, it is important to examine some of the obstacles that still stand in the way of its progress. Ironically, many of these obstacles are related to management and are aggravated by the very characteristics that make AI useful. The purpose of this paper is to heighten awareness of management issues in AI development and to focus attention on their resolution.

2.0 Introduction

For the purpose of this paper, the emphasis is on the subset of AI known as Knowledge Based Systems (KBS). However, much of the discussion can be extrapolated to other areas of Artificial Intelligence (AI) with minor modifications. In order to discuss the future needs of AI technology, it is useful to look at the present state of the art and some trends that have emerged. Over the past few years, a great abundance of small prototype KBSs have been built. These have been developed, using various Knowledge Engineering tools, to solve limited domain problems with reasonable success. However, larger systems that are tackling full scale customer problems have been much slower in coming. This has created a wait and see attitude toward the use of KBS in many customers minds.

Those organizations that are in the forefront of technology and have been developing KB systems, are looking more and more at integrating KBS into the mainstream of computing technology as opposed to stand-alone environments. Thus, an emphasis on total system approaches are emerging. System engineering approaches traditionally look at such things as: life cycle management, documentation, quality, testing, and cost among others. These trends lead to the conclusion that rigorous methodologies are needed for AI development and integration. It is the issues that arise from the desire to apply rigorous methodologies to AI that the rest of this paper is focused on.

3.0 Discussion

There are a number of perspectives of AI that can be examined. One is the view of the AI technologist. This view sees AI as a leading edge technology to be explored as a solution to every problem and developed as an art form. Then there is the view of the old style engineer that rejects AI as a bag of risky tricks. Perhaps the most important view is that of the end customers. They are looking for the solutions to problems at the best overall price. Since the customer is the person paying for the work, system engineers should have their needs in mind when designing and developing systems to solve problems. Let us examine the implications of this last perspective.

Although tremendous strides have been made by micro computers, most organizations use mini and main frame computers for the bulk of their computational problems. These environments typically include in excess of 100,000 lines of software (S/W) code, data bases along with their management systems running in an integrated fashion. Whether being added to or being designed and developed from the ground up, it is these environments that must be considered when looking at the use of AI to solve a significant customer problem. From this perspective, KBS is another piece of S/W that should run on (or with) existing/planned hardware and be integrated with the other functions of the environments. Accepting this premise, let us proceed by examining how KBS development might fit into the mainstream of S/W engineering and what the ramifications might be.

3.1 Typical Software Development Process. Perhaps the most rigorous S/W development methodology is the one developed for Department of Defense programs in the form of DOD Standard 2167A. Most other S/W development life cycles can be extracted as a variation or subset of the DOD one. Figure 3-1 shows a typical S/W development life cycle. Since it is familiar to most, only the briefest description of the phases is given here.

In the Concept Definition phase, the basic ideas for the systems are developed through studies and trade analyses. Here the requirements are developed and documented in the form of top-level system design specifications and operational concepts. These requirements may be allocated to major system components.

In the Preliminary Design phase, the Design Specifications is finalized and a preliminary design is generated. Here the generation of interface and data base specifications takes place along with the development and validation of critical methods (such as algorithms), and test planning. All of which is presented at a Preliminary Design Reviews (PDR).

In the Detailed Design phase, the system is finalized in terms of Detailed Specifications, interfaces, and data base specifications. System test plans are developed along with operations manuals, and prototype testing and simulation takes place. This phase culminates in a Critical Design Review (CDR). The Development phase consists of coding the software according to the designs and specifications. During this phase, detailed test procedures are also developed. The Formal Test phase consists of a hierarchy of test and validation activities. These incrementally test and integrate the system prior to final acceptance and delivery. In the Maintenance phase, the system is kept running properly with occasional corrections and updates as necessary.

3.2 Idealized KBS Development. Most KBS have been developed on a seat of the pants basis. The methodology applied was greatly dependent on the particular building tool being used and on the background of the developers. Presented below is a more formal and somewhat idealized KBS development cycle.

Being with a Problem Identification phase that analyzes the problems and determines which portions are applicable to a KBS solution. In this phase, basic concepts and approaches are developed for the appropriate domains along with a development plan. The key participants and their roles are identified and a cost and benefits analyses the effort is performed.

Next, the Prototype phase develops a full understanding of the domain and task via the building of an initial capability. This prototype is used to develop a detailed design along with performance criteria, test cases, and selection of the tools and target environment. During the Development phase, the prototype is expanded to its full functionality. The user interface is developed and it is converted to fit the target

Figure 3-1. Standard Software Development Life Cycle

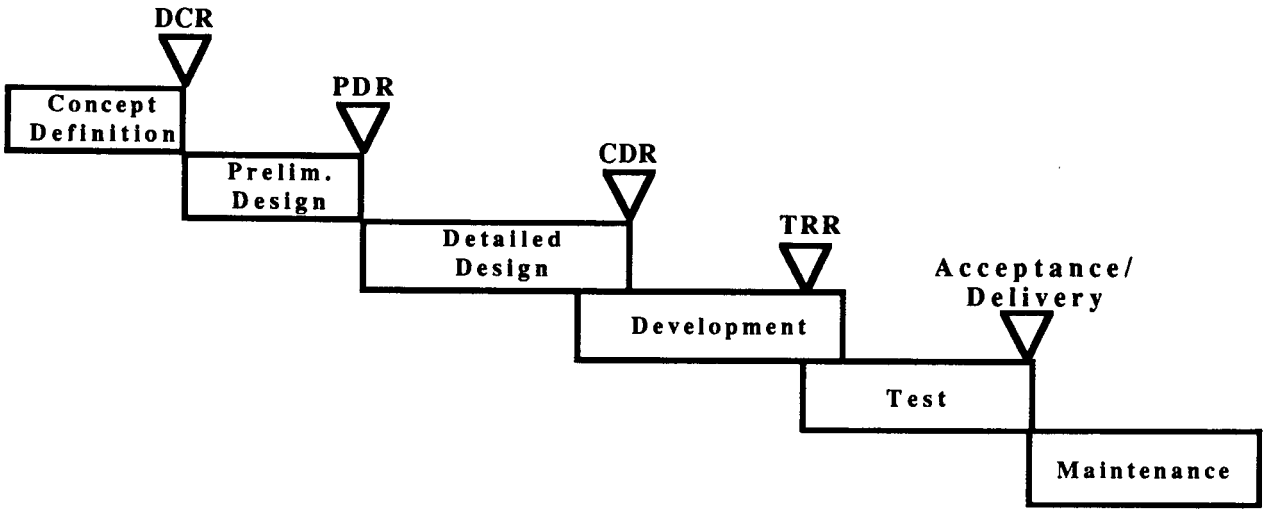
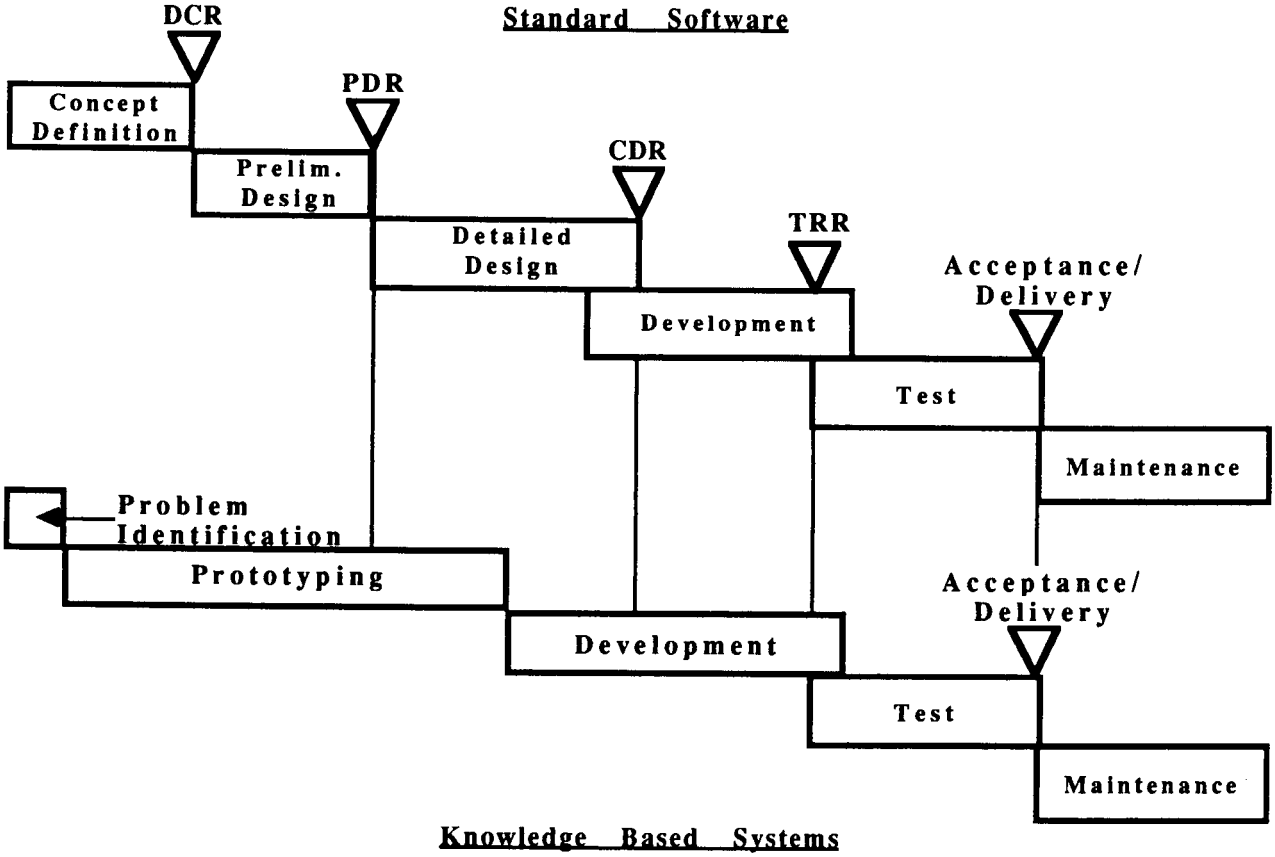


Figure 3-2. Integrated KBS Development in the Standard Software Development Life Cycle



environment. In the Evaluation phase, the system is tested against agreed upon criteria and is operated by experts against new scenarios. Here, if necessary, interfaces to other systems and data bases can take place and system performance can be enhanced before final delivery, documentation and training. In the Maintenance Phase, the system is corrected and updated as necessary for optimal operations.

3.3 An Integrative Methodology. In order to gain full benefits from the Idealized KBS methodology above, it needs to be integrated into a typical S/W development life cycle. Figure 3-2 shows how the two may be overlayed in an integrated development that contained a KBS component imbedded in a larger S/W system.

This proposed combination naturally imposes some of methodology rigor of the standard S/W development cycle onto that of the KBS components. That in turn leads to the examination and discussion of weaknesses that exist today in the management of Knowledge Based Systems.

3.4 Areas to Develop. By systematically examining each phase of a KBS development cycle, and comparing it to the corresponding standard S/W life cycle phase, areas needing development come to light. A top-level cut has been done, and below are some of the areas ripe for further attention and development.

Beginning with the problem Identification phase (in Concept Definition), the first question should be "what are proper applications for KBS solutions?" This is not a trivial question since many problems can be solved by classical methods better than by Knowledge Based ones. Although many simple tests of applicability exist, there are few in depth methodologies widely accepted. It is suggested that such a methodology (exemplified in Figure 3-3) can be developed by assessing: problem characteristics, future role of the subsystem, design characteristics, organization values and impacts, required performance, and operational environments, to name the major areas. Decision trees can be put together by decomposing the above areas to lower levels and adding weights to them as appropriate. Once the methodology is fully developed, it can be calibrated by running it against existing KBS and their associated successes in the field.

In conjunction with the technology selection, process should be a cost/benefit analysis. This might be based on such things as the benefits of replicating an expert, or the savings compared to solving the problem via a different approach. More specifically, the key cost areas for developing a KBS are; knowledge engineering time, domain expert's time, users' time, design, development, and test for prototype and delivered KBS, hardware, and management. The benefits may include; increased productivity, new services or products, elimination of systems or procedures, improvements in quality, fewer or less qualified staff needs, and increase in equipment life. The above factors need to be quantified for each system and traded against each other.

For the sake of this discussion, let us consider a medical advisor application. Such applications are widely accepted as appropriate for a knowledge based system, as opposed to standard software. For this reason, we can expect that any cost/benefit assessment will be favorable.

The next hurdle during Concept Definition (early prototyping) is to characterize the requirements/specifications. One method, shown in Figure 3-4, is to capture the functional requirements and decompose them as far as practical. Once a decomposition exists, the relationships between functions can be developed. This process is important because it will highlight the boundaries to the applicability of of KBS solutions. Taking the perspective that some parts of a problem can be straightforward and solvable by conventional methods may reduce the complexity of the KBS

Figure 3-3. Selection Methodology for KBS Applications

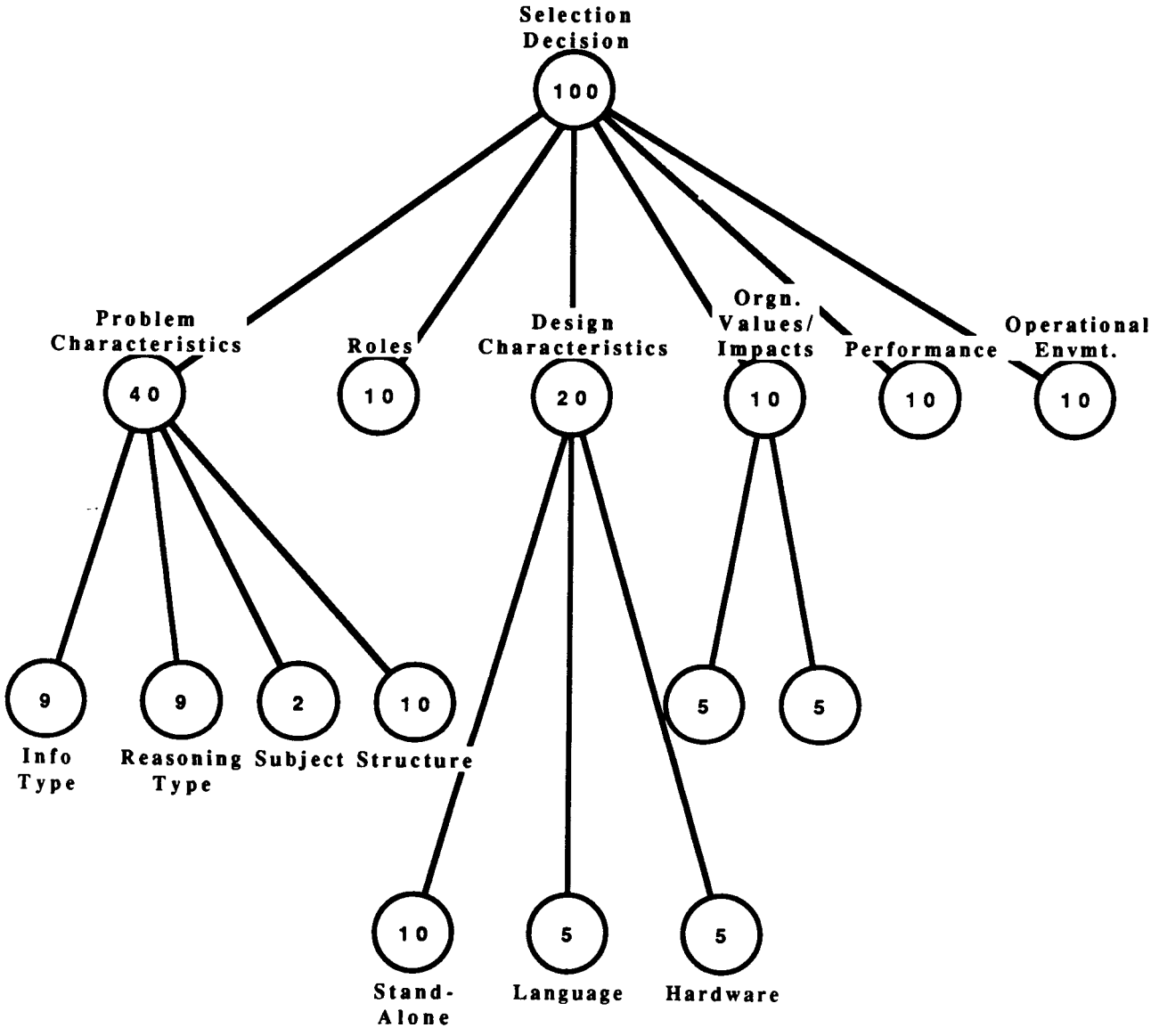
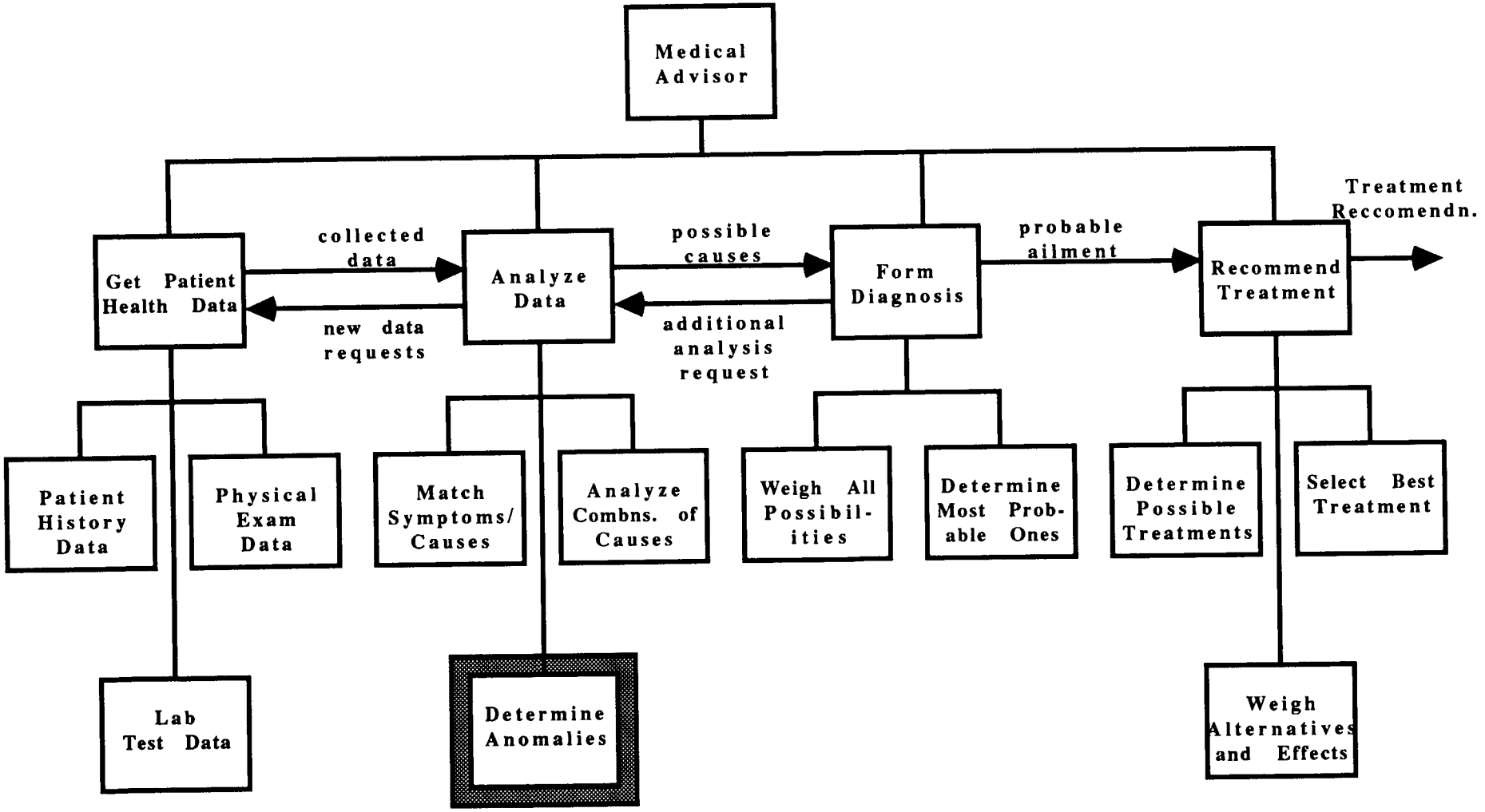


Figure 3-4. Sample Functional Decomposition for a KBS



components. One example is the "Determine Anomalies" function in Figure 3-4. This may be accomplished by a conventional database-driven limit checking program. Another example is the area of scheduling problems. Some solutions can be broken up into heuristic and algorithmic components. In addition to functional specifications, performance specifications should be developed for KBS. These a re important since they affect the hardware and software selections made for the systems. The most important performance characteristics can be along the lines of quality, quantity, speed, or interfaces. In the medical advisor example, quality can be specified in terms of the percentage of correct diagnoses and/or recommendations. Quantity or speed can be assessed in terms of how quickly (once the required input data is provided) a diagnosis or recommendation can be completed. Finally, the interfaces should be specified in terms of the user's needs and the delivery environment. In this case a doctor may be the user, with an interactive interface as well as access to a database of patient history information, and traditional software to perform initial anomaly screening.

As the development of a system moves through the design phases, much emphasis is placed on documentation and reviews. In the case of KBS, documentation standards and review milestones are not well defined. The establishment of Preliminary and Critical Design Review (PDR, CDR) milestones for KBS should be considered an important part of system management. They are valuable for the developers to work toward, and for the customers/reviewers to understand the product being developed and gain confidence in its capability.

At PDR the knowledge gained from prototyping should be presented as it relates to the full KBS. The detailed functionality and performance requirements of the KBS should be documented. Some possibilities for the documentation are: functional decompositions, logic networks, decision flows, interface diagrams, and operational concept descriptions. Naturally, some of the KBS development tool outputs can be used for documentation, but most are usually more appropriate for a CDR level review. At CDR, the focus should be on the full integrated system. The knowledge base should be complete and captured by the development tool. Text level documentation should be provided describing interfaces between the KBS and other system components. Full screen level user interfaces should be defined and presented. There is a considerable amount of work still to be done in the area of KBS documentation standards.

A final area to be addressed here is that of testing and validation of KBS. This is perhaps their most important and also weakest area from the perspective of traditional operational systems. It is important because operational systems need to be trusted by users. Failure of a $300M dollar satellite, the life support system of a space vehicle, or your doctor's diagnostic advisor is likely to be viewed by users as not merely undesirable, but catastrophic. If a KBS is to be a key component of successful operation, then users will demand a high degree of confidence in their reliability. The AI community is of several opinions with regard to methods for testing KBS, ranging from the position that testing KBS is no different from testing any other software system, to a belief that KBS are so different that present methods cannot be applied to them. Nevertheless, major KBS have been built and put into operation successfully using variants of traditional; methods. Development of a reliable methodology for verification and validation of KBS must still be considered a high priority issue from the management perspective.

Some approaches can be outlined for achieving the required level of reliability in KBS. One is to explore the use of multiple concurrent KBS to provide redundancy in critical applications. This can be done through the use of multiple copies of the KBS in a voting arrangement, such as is used with traditional software systems in the shuttle today. Another is to develop multiple KBS addressing the same application, but drawing on different knowledge sources to cross validate each other.

A third approach is to develop the same KBS using different development tools. This latter approach is more cost effective than the second because knowledge acquisition (typically the most costly activity in KBS development) is done only once.

4.0 Conclusion

There are other areas of the software development life cycle which give rise to problems when applied to KBS. As the technology matures, and the issues discussed here are resolved, other issues will become more important. However, progress toward resolution of these issues, and development of a methodology for building KBS, will be crucial in gaining the support by management required for KBS technology to be integrated with traditional systems in operational environments.